



# Inhalt

<b>Der Objekt Relationale Force4 E/R Modeler</b>	<b>2</b>
<b>Übersicht über die Force4-Modeler Funktionen</b>	<b>2</b>
Model	2
Entitäten und Attribute	2
Relationen	3
Primärschlüssel-Generierung	3
<b>Funktionalitäten</b>	<b>4</b>
Reverse Engineering	5
Model-Import	6
Model-Export	7
Force4-Serverkomponenten	8
DDL-Script	9
Allgemeine Funktionalitäten auf Entitäten-, Attribut- und Relationen-Ebene	11
<b>Beschreibungen der Bearbeitungsmasken</b>	<b>12</b>
Model	12
Entitäten	14
Attribute	15
Relationen	18
Primärschlüssel-Generierung	20

## Der Objekt Relationale Force4 E/R Modeler

ist ein E/R-Modeling-Tool mit dem man auf einfache Art Datenstrukturen in einem Repository hinterlegt, um sie während der Modellierungs-Phase eines Prozesses einfach und schnell auf dem aktuellen Stand zu halten. Weiterhin fungiert der Force4-Modeler als Entwicklungs-Schnittstelle zwischen der Datenbank und JAVA respektive Macromedia Flex mit den jeweiligen spezifischen Nomenklaturen.

Das Model im Force4-Modeler ist zunächst ein abstraktes "Gebilde", bestehend aus Entitäten, Attributen und Relationen, um die Datenstrukturen aufzubauen und darzustellen, die sich zum Beispiel aus einem Business-Prozess ergeben. Der Aufbau kann manuell im Modeler vollzogen werden, das Einlesen von Strukturen aus bestehenden Datenbanken durch direkte Verbindung via JDBC, dem sog. Reverse Engineering ist auch möglich.

Ist ein Model entsprechend den funktionalen Vorgaben erstellt worden, so lassen sich damit zum einen Datenbanken, zum anderen aber auch direkte Schnittstellen zu JAVA und Macromedia Flex erstellen.

Entitäten und Attribute sind die abstrakten Gegenstücke zu Tabellen und Tabellenspalten in einer Datenbank. Zusätzlich zu den Eigenschaften, die notwendig sind, um eine Datenbank-Tabelle zu definieren, lassen sich in Entitäten und Attributen weitere Eigenschaften einstellen, um die Kompatibilität zu JAVA- und Macromedia Flex zu erzeugen.

Relationen im Force4-Modeler sind die Verbindungen zwischen den Entitäten. Durch den Aufbau dieser Relationen wird aus einer Ansammlung von Entitäten eine relationale Struktur. Die Abbildung der Relationen auf die Datenbank zeigt sich durch den Aufbau von sog. Foreign Key Constraints, Bedingungen auf Datenbank-Ebene, die die Beziehungen der Tabellen untereinander widerspiegeln (soweit die Datenbank dies unterstützt).

## Übersicht über die Force4-Modeler Funktionen

### Model

Ein Model benötigt zur Identifizierung immer einen Namen und das so genannte "Connection Dictionary" welches aus der Verbindungs-URL des JDBC-Adapters mit seinen Login-Daten und eventuell zusätzlich benötigter Treiber und Plugins besteht.

Das Connection Dictionary besteht wiederum aus den Treiber-Informationen, die speziell für die ausgewählte Datenbank und JDBC-Adapter gelten wie auch die Datentypen, die es für diese Datenbank gibt. Die Datentypen lassen sich dann auf Attribut-Ebene setzen und haben bei der Generierung der DDL-Scripte Relevanz.

Der Vorteil des Force4-Modelers ist die Unabhängigkeit des Models von seinem aktuell zugewiesenen Adapters zur Datenbank. Durch Änderung der Adapter-Daten findet eine Analyse des neuen Adapters und eine automatische Anpassung der Datentypen statt. Das manuelle Anpassen der Model-Daten ist somit nur im Einzelfall vonnöten.

Die Verwaltung von Datenmodellen im Force4-Modeler ist durch Funktionen wie Duplizieren, Model-übergreifendes Kopieren von Entitäten, Attributen und Relationen, einfache Änderungen der Adapter, Import und Export effizient und durch die intuitive Benutzerführung mit sehr geringem Aufwand verbunden.

### Entitäten und Attribute

Die Tabellen-definierende Entität mit seinen Attributen bildet als inhaltsgebende Komponente die Basis für eine Datenstruktur. Neben den Namen und Randparametern für die externe Repräsentation der Struktur für die Datenbank (externer Name, externer Datentyp) werden interne Eigenschaften gepflegt, die die Konnektivität in Richtung JAVA und Macromedia Flex bestimmen (interner Datentyp, interner Name, Sichtbarkeit eines Attributes, Berechtigungen) und Eigenschaften die beide Repräsentationen betreffen (Primärschlüssel, NULL-Werte, Spaltendefinition)







## Relationen

Aufbauend auf den Entitäten eines Modells wird die Datenstruktur durch die verbindenden Elemente, den Relationen vervollständigt. In einem relationalen Datenbank-Modell hängen Tabellen über sog. Master-Detail-Relationen zusammen. Dabei gibt es z.B für einen Master-Datensatz viele Detail-Datensätze, zusammengehalten über eine 1/n-Relation (1 Master/ n (viele) Details), die im allgemeinen von der Primärschlüssel-Spalte der Master-Tabelle auf eine Fremdschlüssel-Spalte der Detail-Tabelle zeigt. Im Force4-Modeler lassen sich diese Relationen zwischen Entitäten erzeugen. Die datenbankspezifische Umsetzung und die Generierung in Richtung JAVA und Macro-media Flex wird vom Force4-Modeler übernommen.

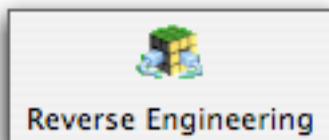
## Primärschlüssel-Generierung

Neben der Möglichkeit die Primärschlüssel-Unterstützung der Datenbank über das DDL-Script zu erzeugen und im Standard zu verwenden kann der Force4-Modeler eine datenbankunabhängige Primärschlüssel-Unterstützung verwalten und in Richtung JAVA generieren. Auf den Ebenen Model, Entität und Attribut lassen sich die Methodiken, wie auch die Parameter dieser Methodiken global aber auch spezifisch auf jeder Ebene setzen.

## Funktionalitäten

<b>Allgemeine Funktionalitäten auf Model-Ebene</b>	
	
	<p><b>Neu</b> Erstellen eines neuen leeren Models, entweder durch Bereitstellung eines neuen A-dapters oder durch Auswahl eines vorhandenen JDBC-Treibers.</p>
	<p><b>Bearbeiten</b> Bearbeiten des aktuell ausgewählten Models</p>
	<p><b>Löschen</b> Löschen des aktuell ausgewählten Models</p>
	<p><b>Duplizieren</b> Duplizieren des aktuell ausgewählten Models</p>
	<p><b>Voreinstellungen</b> Modeler-spezifische Voreinstellungen für Reverse Engineering und DDL-Script-Erzeugung</p>

## Reverse Engineering



Erstellen eines Models aus einer bestehenden Datenbank über eine JDBC-Verbindung

Schritt 1: Um ein Reverse Engineering eines Models aus einer Datenbank vornehmen zu können, müssen folgende Informationen bekannt sein: die Verbindungs-URL, um über JDBC die Datenbank zu erreichen, der Benutzername und das Passwort für die Datenbank-Verbindung ,eventuell zusätzliche JDBC-Treiber oder Plugins für den verwendeten JDBC-Connect.

Schritt 2: Auswahl der zu engineerenden Schemas und Datenbank-Objekt-Typen mit der Möglichkeit die Anzahl der Datenbank-Objekte durch ein Suchpattern einzugrenzen und die Möglichkeit Stored Procedures mitzuladen. Diese Informationen kommen direkt von der Datenbank, die über den JDBC-Adapter abgefragt wurde.

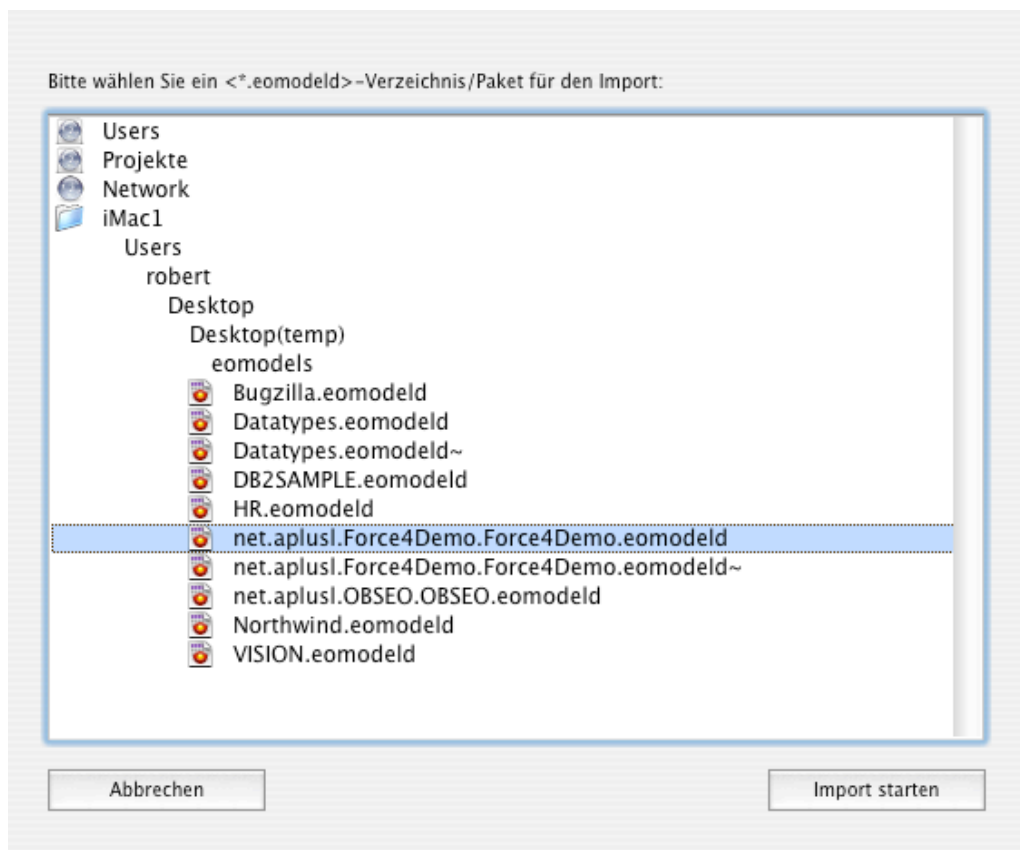
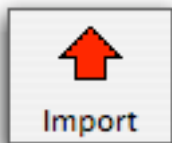
Schritt 3: Auswahl der zu engineerenden Datenbank-Objekte und Procedures: Ein Model kann aus allen Elementen der Datenbank oder nur aus ausgewählten Teilbereichen bestehen.

Schritt 4: Auswahl der Reverse Engineering Einstellungen

Reverse-Engineering Einstellungen	
<input checked="" type="checkbox"/>	<b>Inverse Relationen einbinden</b> Die in der Datenbank vorliegenden „ToMany“-Relationen werden in das Model übernommen
<input checked="" type="checkbox"/>	<b>Relationen einbinden</b> Die in der Datenbank vorliegenden „ToOne“-Relationen werden in das Model übernommen.
<input checked="" type="checkbox"/>	<b>Tabellennamen mit Schemanamen erzeugen</b> Die Tabellen-Namen werden in der Form <SCHEMA.TABELLE> in die Entitäten-Spalte „Externer Name“ eingelesen
<input checked="" type="checkbox"/>	<b>Attribute als &lt;sichtbar&gt; einlesen</b> Alle Attribute einer Entität werden als sichtbar in das Model eingelesen, unabhängig ob sie in der Datenbank sichtbar sind oder nicht.
<input checked="" type="checkbox"/>	<b>Entitäten Klassenbezeichnungen auslesen</b> Es werden Model-bezogene Klassenbezeichnungen für die Entitäten erzeugt.
<input checked="" type="checkbox"/>	<b>Minimale Anzahl an Locking-Attributen verwenden</b> Es werden die Entitäten mit einer minimalen Anzahl an Locking-Attributen (Attribute, die an einer Update-WhereClause beteiligt sind) eingelesen, im Normalfall ist dann nur der Primärschlüssel als Locking-Attribut markiert

## Model-Import

Importieren eines Modells welches im Filesystem liegt (im allgemeinen in der Form eines „\*.eomodeld“-Paketes oder -Verzeichnisses)

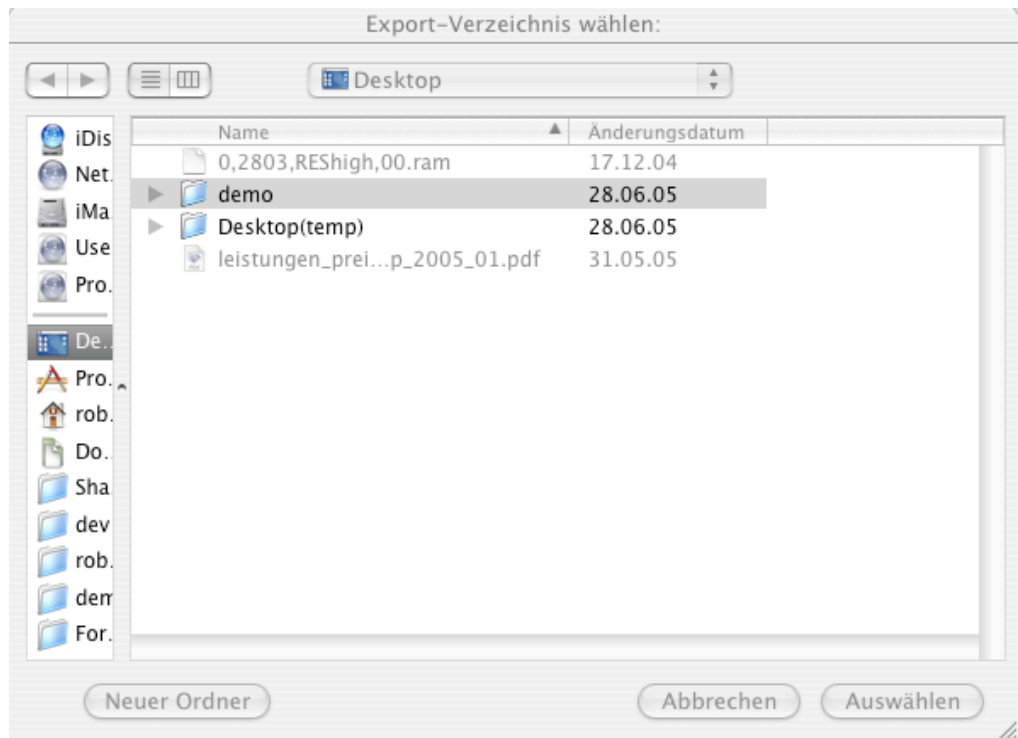
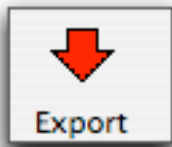


Sollte der Modelname, der in der Index-Datei des zu importierenden Modells hinterlegt ist, schon im System vorhanden sein, erscheint dieser Hinweis mit drei Möglichkeiten des Fortfahrens: Abbruch, Neue Bezeichnungen (Modelname, Paketname) vergeben oder bestehendes Model ersetzen.

Entscheidet man sich in Schritt 2 zur Vergabe neuer Namen so erscheint zuerst die Aufforderung zur Eingabe eines neuen Namens und als zweites die Aufforderung zur Eingabe eines neuen Paketnamens.

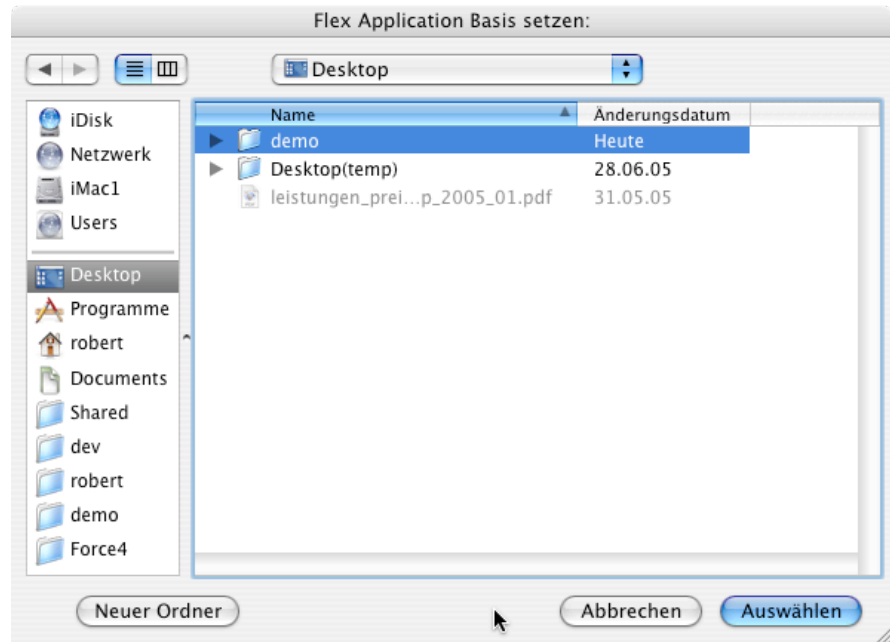
## Model-Export

Export des aktuell ausgewählten Modells ins Filesystem



## Force4-Serverkomponenten

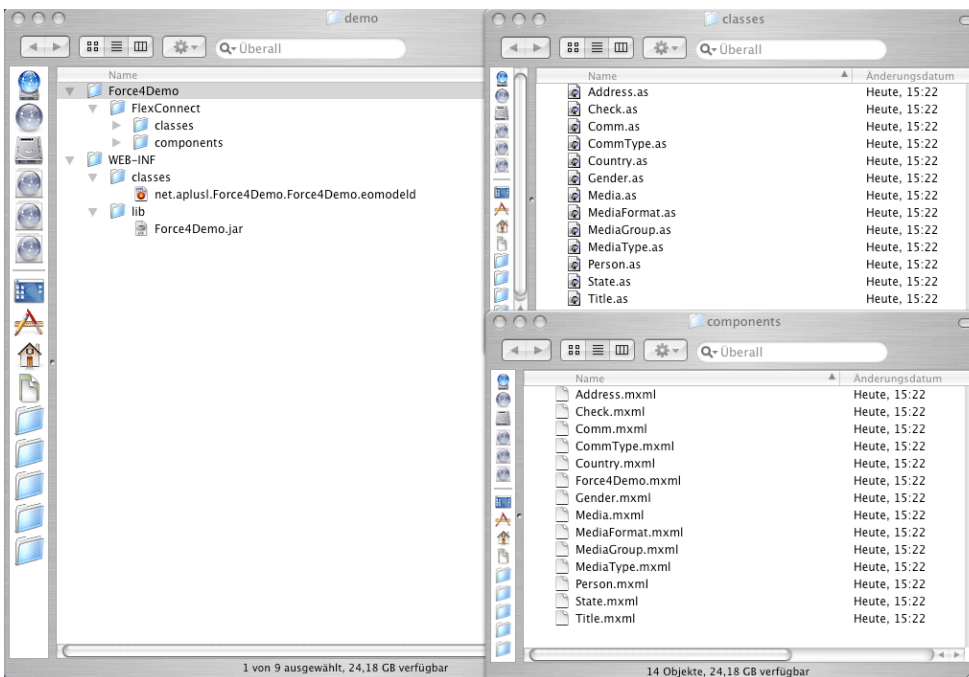
Erstellen der für Macromedia Flex und die für den Force4-Server wichtigen Komponenten des aktuell ausgewählten Modells



Server Components – Schritt 1: Filedialog zur Auswahl des Force4/Flex-Servers. Voreingestellt sind in diesem Filedialog die in den Composer-Voreinstellungen und, bei Verwendung eines alternativen Force4-Servers, im Model hinterlegten Basis-Verzeichnisse für Flexserver und Force4-Server.

Server Components – Schritt 2: Aufforderung zum Stoppen des Force4-Servers. Um den JAVA-Anteil der Force4-Server-Komponenten zu generieren, muss der Server in dessen Verzeichnis generiert wird gestoppt werden.

Nach erfolgreicher Generierung erfolgt ein Hinweis mit der Bitte, den vorher gestoppten Force4-Server wieder zu starten.

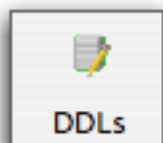


ServerComponents – Ergebnis: Blick ins Filesystem auf die generierten Dateien



## DDL-Script

Erstellen von DDL-Scripten des aktuell ausgewählten Modells mit der Möglichkeit des Speicherns ins Filesystem und der direkten Ausführung



Tabellen löschen  
 Primärschlüssel-Unterstützung löschen

Fremdschlüssel erzeugen  
 Primärschlüssel erzeugen  
 Primärschlüssel-Unterstützung erzeugen

```

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE, LOCKING PESSIMISTIC;

-- The 'Drop Primary Key Support' option is unavailable.;

DROP TABLE "ADDRESS" CASCADE;
DROP TABLE "XCHECK" CASCADE;
DROP TABLE "COMM" CASCADE;
DROP TABLE "COMM_TYPE" CASCADE;
DROP TABLE "COUNTRY" CASCADE;
DROP TABLE "GENDER" CASCADE;
DROP TABLE "MEDIA" CASCADE;
DROP TABLE "MEDIA_FORMAT" CASCADE;
DROP TABLE "MEDIA_GROUP" CASCADE;
DROP TABLE "MEDIA_TYPE" CASCADE;
DROP TABLE "PERSON" CASCADE;
DROP TABLE "XSTATE" CASCADE;
DROP TABLE "TITLE" CASCADE;
        
```

DDL ausführen

In Datei speichern

Schließen

DDL Script: Für das aktuell ausgewählte Model wird hier das Script zur Erstellung der dazugehörigen Datenbank erzeugt.

Mit "DDL ausführen" wird das aktuell sichtbare Script über den JDBC-Adapter gegen die Datenbank abgesetzt

DDL Optionen	
<input checked="" type="checkbox"/>	<b>Tabellen löschen</b> Ist diese Option ausgewählt, so werden die Tabellen vor dem Erstellen gelöscht
<input checked="" type="checkbox"/>	<b>Primärschlüssel-Unterstützung</b> Für Datenbanken, die dies unterstützen, wird die Primärschlüssel-Unterstützung (z.B. löschen von Sequences für Oracle Datenbanken) vor Erstellen der Tabellen gelöscht

DDL Optionen	
<input checked="" type="checkbox"/>	<b>Primärschlüssel erzeugen</b> Es werden Primary Key Constraints auf die Tabellenspalten gelegt, die im Model mit dem Primärschlüssel-Attribut versehen sind.
<input checked="" type="checkbox"/>	<b>Fremdschlüssel erzeugen</b> Es werden die, sich aus den Relationen ergebenden, Foreign Key Constraints erzeugt
<input checked="" type="checkbox"/>	<b>Primärschlüssel-Unterstützung erzeugen</b> Für Datenbanken, die dies unterstützen, wird die datenbankseitige Unterstützung für die Generierung von Primärschlüsseln (z.B. Sequences für Oracle Datenbanken) erzeugt.

## Allgemeine Funktionalitäten auf Entitäten-, Attribut- und Relationen-Ebene

Dieses Kontextmenü findet sich in der Entitäten-/ StoredProcedure-Liste, der Attribut-/ Parameter-Liste und der Relationenliste. Die Menüpunkte korrelieren in etwa mit der Knopfleiste an der rechten Seite der Bearbeitungsfenster:

<b>Hinzufügen</b>	⌘N
Kopieren	⌘C
Einfügen	⌘V
Duplizieren	⌘D
Entfernen	⌘X
Sichern	⌘S
Verwerfen	⌘Z

<p><b>Hinzufügen/Neu</b> Eine neues Objekt (je nach Fenster Entität/Attribut/...) wird angelegt.</p>
<p><b>Kopieren</b> Aktuelles Objekt in den Force4-Zwischenspeicher (dieser Punkt existiert nur im Menü)</p>
<p><b>Einfügen</b> Aktuellen Force4-Zwischenspeicherinhalt in Liste einfügen (dieser Punkt existiert nur im Menü)</p>
<p><b>Duplizieren</b> Aktuelles Objekt duplizieren</p>
<p><b>Entfernen</b> Aktuelles Objekt entfernen</p>
<p><b>Sichern</b> Aktuellen Stand sichern</p>
<p><b>Verwerfen</b> Zurückkehren zu letztem gesicherten Stand</p>

## Beschreibungen der Bearbeitungsmasken

### Model

Eigene Bezeichnung  
Force4Demo

Modellname  
Force4Demo

Paketname  
net.aplus.Force4Demo

JDBC Treiber

JDBC-Plugin

Verbindungs-URL  
jdbc:FrontBase://koloss.a-line.de/Force4Demo?isolation=repeatable\_read/locking=optimistic

Initiales SQL Statement

Primärschlüssel-Generierung  
Globales Verfahren Spez  
Default  
Parameter  
Wert

Benutzername  
force4

Kennwort  
\*\*\*\*

Flex Server als Force4 Server verwenden  
 alternativen Force4 Server verwenden

Aktueller JDBC Treiber  
FrontBase - FBJDriver - 2.3

#### Eigene Bezeichnung

Die sichtbare Modellbezeichnung im Force4-System. Sie zeigt sich im Modeltree und im Composer zur Feldauswahl

#### Modellname

Die Bezeichnung des Modells für die Server-Representation, der offizielle Modellname, der in Verbindung mit dem Paketnamen für die Flex- und Java-Generierung verwendet wird. Er unterliegt einigen Restriktionen in Bezug auf die Zeichenauswahl. Erlaubt sind alphanumerische Zeichen wobei der Anfang des Namens nicht aus einer Ziffer bestehen darf

<b>Paketname</b> Zusammen mit dem Modelnamen bildet der Paketname eine im System eindeutige Bezeichnung. Diese wird für die Identifizierung des Models im System benötigt. Ist der Modelname im System schon eindeutig, so kann der Paketname auch leer gelassen werden. Gängige Paketnamen (analog zu JAVA) sind z.B. die Kombination der inversen Domain und des Modelnamens (net.aplusl.Force4Demo). Jeder andere Name ist aber auch möglich, solange er zusammen mit dem Modelnamen eine Eindeutigkeit im System erzeugt
<b>JDBC Treiber</b> Wird für den JDBC-Adapter ein zusätzlicher Treiber benötigt, so kann der Treibername hier hinterlegt werden. Im Normalfall reicht der Standard-Treiber aus, auf den die Verbindungs-URL zeigt
<b>JDBC-Plugin</b> Wird für den JDBC-Adapter ein zusätzliches Plugin benötigt, so kann der Plugin-Name hier hinterlegt werden. Im Normalfall reicht der Standard-Treiber aus, auf den die Verbindungs-URL zeigt.
<b>Benutzername</b> Der Datenbank-Benutzername mit dem der JDBC-Adapter sich an die Datenbank anmelden soll
<b>Kennwort</b> Das Kennwort des Datenbank-Benutzers mit dem der JDBC-Adapter sich an die Datenbank anmelden soll.
<b>Verbindungs-URL</b> Die JDBC-Verbindungs-URL enthält neben dem datenbankspezifischen JDBC-String (jdbc:<datenbankconnect>: ) Angaben über den Datenbank-Host und die Datenbank. Z.B. für Oracle: „jdbc:oracle:thin:@ORACLE-HOST:ORACLE-PORT:ORACLE-SID“ oder für FrontBase: „jdbc:FrontBase://FB-HOST/FB-DBName/“
<b>Initiales SQL Statement</b> Benötigt der JDBC-Adapter direkt nach dem Anmelden an die Datenbank weitere SQL-Statements um die Session zu initialisieren, so können sie hier hinterlegt werden.
<b>Flex Server als Force4 Server verwenden</b> Der Macromedia Flex Server hat eine eigenen JAVA-Server, der im Standard verwendet wird. Die von Force4 verwendeten Dateien werden in die Flex Server- Struktur generiert. Die Flex-spezifischen MXML- und ActionScript-Dateien werden unabhängig davon immer in die Flex-Architektur integriert.
<b>Alternativen Force4 Server verwenden</b> Hat man einen eigenen Java-Server zur Ausführung der Force4-Server-Komponenten, so kann man ihn unter diesem Punkt einstellen
<b>Force4 Server Wurzelverzeichnis</b> Verwendet man einen alternativen Force4-Server, so ist dies das Force4-Wurzelverzeichnis, in das die Server-Komponenten generiert werden.
<b>Force4 Server Endpoint URL</b> Ein alternativer Force4 Server benötigt eine URL, die auf den so genannten Endpoint des Servers zeigt.

## Entitäten

13 Entities in Model Force4Demo				
Name		Externer Name	Einschränkende Bedingung	Batch Fehler Anzahl
Address		ADDRESS		
Check		XCHECK		
Comm		COMM		
CommType		COMM_TYPE		
Country		COUNTRY		
Gender		GENDER		
Media		MEDIA		
MediaFormat		MEDIA_FORMAT		
MediaGroup		MEDIA_GROUP		
MediaType		MEDIA_TYPE		
Person		PERSON		
State		XSTATE		
Title		TITLE		

Entität: Address				
Address		<input type="checkbox"/> Schreibgeschützt	ADDRESS	0
Primärschlüssel-Generierung		Externe Abfrage		
Globales Verfahren	Default			
Spezifisches Verfahren	Default			
Parameter				
Wert				

### Name

Name der Entität; muss eindeutig im Model sein; darf nicht wie das Model oder eine ihrer Relationen heißen.

### Schreibgeschützt

Ist das Häkchen gesetzt, so ist diese Entität nur lesend erreichbar. Einfügen, löschen und aktualisieren ist nicht erlaubt.

### Externer Name

Name der externen Repräsentation der Entität; im allgemeinen der Tabellen-Name der dazugehörigen Datenbank-Tabelle

### Einschränkende Bedingung

Auf Felder der zur Entität zugehörigen Tabelle kann eine einschränkende Bedingung gesetzt werden, die beim selektieren der Entität berücksichtigt wird. Z.B. mandantenabhängiges Selektieren (ORGANISATION\_ID = 4711).

### Batch Fehler Anzahl

Anzahl der gleichzeitig übermittelten Einzelfehlermeldungen im Falle eines Scheiterns der Abfrage. Das Nachladen von Folgefehlermeldungen erzeugt Last auf der Leitung zum Adapter, die somit minimiert werden kann. Die Auswertung der gesammelten Fehlermeldungen ist damit in die Business-Logik verschoben.

### Externe Abfrage

Es kann eine externe Abfrage zur Entität gesetzt werden, die aufgerufen wird, soweit es für diese Entität eine unqualifizierte Anfrage gibt: Beispiel: die Entität wird ohne Bedingung befragt. Anstelle von

„SELECT \* FROM ENTITY“ kann jetzt hier hinterlegt werden:

„SELECT NAME, PHONE, STREET, TOWN FROM ENTITY WHERE ORG\_ID=4711“

## Attribute

Name	Externer Name	Interner Typ	Externer Typ	Width	Precision	Scale
checkid	CHECK_ID	Integer (i)	INTEGER		31	
city	CITY	Character	VARCHAR	1000		
companyName	COMPANY_NAME	Character	VARCHAR	300		
countryId	COUNTRY_ID	Integer (i)	INTEGER		31	
id	ID	Integer (i)	INTEGER		31	

### Name

Name des Attributes, muss eindeutig in der Entität sein, darf nicht wie eine Relation der Entität heißen.

### Primärschlüssel

Ist das Häkchen gesetzt, so ist dieses Attribut ein Primärschlüssel

### Sichtbar

Ist das Häkchen gesetzt, so ist dieses Attribut sichtbar für die Verarbeitung

### Update Snapshot Attribut

Ist das Häkchen gesetzt, so wird dieses Attribut im Falle einer Aktualisierung des Datensatzes zur Identifizierung in der UPDATE-WhereClause verwendet

### Null erlaubt

Ist das Häkchen gesetzt, so ist „NULL“ als Wert für dieses Attribut erlaubt

### Schreibgeschützt

Ist das Häkchen gesetzt so ist dieses Attribut nur lesend erreichbar. Wert-Änderungen sind nicht erlaubt.

### Externer Name

Name der externen Repräsentation des Attributes, im allgemeinen der Spalten-Name der dazugehörigen Datenbank-Tabelle

### Interner Typ

Zur internen Verarbeitung der Attribute in Enterprise Objects, Java und Macromedia Flex gibt es den internen Datentyp. Dieser setzt sich aus einem Basis-Typ (Java Class) und einem Subtyp (valueType) zusammen. Folgende Werte sind möglich:

Interner Typ	Java Class	Value Type	
Character	java.lang.String		Der Standard-Type Character ohne valueType validiert automatisch den Spalteninhalt auf Länge gegen die maximale Anzahl an Zeichen für ein varchar-Feld der Datenbank und wählt die JDBC-Methode „setString“, wenn der Inhalt kürzer als diese ist, „setCharacterStream“, wenn er länger ist.
Character (S)	java.lang.String	S	verwendet immer die JDBC-Methode „setString“

Character (C)	java.lang.String	C	verwendet immer die JDBC-Methode „setCharacterStream“
Character (E)	java.lang.String	E	konvertiert den Text in rohes UTF-8 und sendet diese über die Methode „setBinaryStream“ an ein Binärfeld in der Datenbank. Es wird nicht empfohlen, diesen Typ zu wählen, da es im allgemeinen die Aufgabe der Datenbank ist, die Texte im richtigen Format abzulegen
Character (c)	java.lang.String	c	befreit den Text von allen nachlaufenden Leerzeichen. Dies ist wichtig für den externen Datentyp CHAR, welcher Leerzeichen verwendet um die Spalten aufzufüllen und sollte verwendet werden, wenn diese Leerzeichen ansonsten keine Bedeutung haben
Integer(i)	java.lang.Number	i	repräsentieren die Java-Subtypen java.lang.Integer, java.lang.Byte, java.lang.Double, java.lang.Short, java.lang.Float und java.lang.Boolean über die abstrakte Klasse java.lang.Number.
Byte	java.lang.Number	b	
Double	java.lang.Number	d	
Short	java.lang.Number	s	
Long	java.lang.Number	l	
Float	java.lang.Number	f	
Boolean	java.lang.Number	c	
Integer (B)	java.lang.Number	B	verwendet java.lang.Integer wie Integer(i), jedoch mit einem valueType von B und der Möglichkeit einer größeren Precision (z.B. von 31)
BigDecimal	java.math.BigDecimal		sind die Typen für Dezimalzahlen mit der Möglichkeit Nachkommastellen via „Scale“ zu definieren. Je nach JDBC-Adapter haben sie den valueType B oder keinen valueType zugewiesen.
BigDecimal(B)	java.math.BigDecimal	B	keinen valueType zugewiesen.
Timestamp	com.webobjects.foundation.NSTimestamp		Der Timestamp ohne valueType verwendet die Standard JDBC-Methoden „setObject“ und „getObject“ um Daten zu transferieren. Es wird davon ausgegangen, dass die Datenbank ein Format vorhält welches kompatibel zum verwendeten java.sql.timestamp ist.
Timestamp(D)	com.webobjects.foundation.NSTimestamp	D	verwendet die Methoden „setDate“ und „getDate“ im „java.util.Date“-Objekt
Timestamp(t)	com.webobjects.foundation.NSTimestamp	t	verwendet die Methoden „setTime“ und „getTime“ im „java.sql.Time“-Objekt
Timestamp(T)	com.webobjects.foundation.NSTimestamp	T	verwendet die Methoden „setTimestamp“ und „getTimestamp“ im „java.sql.Timestamp“-Objekt
Timestamp(M)	com.webobjects.foundation.NSTimestamp	M	wird anstelle von Timestamp(D) für MS SQLServer verwendet. Nur das „java.sql.Date“-Objekt wird unterstützt.
Binary Data	com.webobjects.foundation.NSData		

### **Externer Typ**

Diese Auswahlliste trägt alle externen Datentypen, die man über den JDBC-Adapter an die Datenbank übermitteln kann. Die Datentypen sind Datenbank-abhängig und variieren somit von Adapter zu Adapter. Ändert man den Adapter eines Modells, so werden diese Typen auf die neue Datenbank angepasst.

### **Width**

Die maximale Anzahl an ASCII-Zeichen für ein Character-Attribut kann hier hinterlegt werden

### **Precision**

Gesamtanzahl an Stellen in einem numerischen Attribut

### **Scale**

Anzahl der Nachkomma-Stellen in einem numerischen Attribut

### **Ausgabe Format**

Inhalte von Attributen können hier per default schon auf Adapter-Ebene zur Anzeige umgerechnet werden. Dazu wird der Platzhalter „%P“ verwendet. Ist z.B. ein Spalteninhalt eine Dauer, die in Sekunden abgelegt ist, so kann man sie mit dem Ausgabe-Format „%P /60“ direkt in Minuten anzeigen lassen.

### **Eingabe Format**

Inhalte von Attributen können vorab schon auf Adapter-Ebene zum Speichern umgerechnet werden. Dazu wird der Platzhalter „%V“ verwendet. Ist z.B. ein Spalteninhalt eine Dauer, die in Sekunden abgelegt ist, so kann man den Wert mit dem Eingabe-Format „%V \*60“ als Minute eingeben. Gespeichert wird der Sekunden-Wert.

## Relationen

2 Relationen in Entität Comm						
Name		Löschregel	Join Typ	Quell Attribut	Ziel Attribut	
commType		nullify	inner	commTypeld	id ( CommType )	
person		nullify	inner	personId	id ( Person )	

Relation: commType						
commType		<input checked="" type="checkbox"/> Sichtbar	nullify	inner	commTypeld	CommType
Batch Fehler Anzahl		<input type="checkbox"/> Pflichtfeld	cascade	full outer	entry	id
0		<input type="checkbox"/> ToMany Relation	deny	left outer	id	name
		<input type="checkbox"/> Besitzt Ziel	no action	right outer	personId	
		<input type="checkbox"/> Primärschlüssel propagieren				

### Name

Name der Relation, muss eindeutig in der Entität sein, darf nicht wie ein Attribut der Entität und nicht wie die Entität heißen.

### Sichtbar

Relationen sind per default als sichtbar gesetzt. Unsichtbare Relationen werden in JAVA nicht instanziiert und können nicht für Verarbeitungen verwendet werden.

### Pflichtfeld

Ist eine Relation ein Pflichtfeld, so muss Quell-Datensatz und Ziel-Datensatz immer mit gültigen Werten gefüllt sein, d.h. NULL-Werte sind nicht erlaubt.

### ToMany Relation

In einer ToOne-Relation gibt es genau einen Datensatz auf der Quell-Seite und auf der Zielseite der Relation. In einer ToMany-Relation kann es für einen Quell-Datensatz einen, mehrere oder keinen Datensätze auf der Zielseite geben.

### Besitzt Ziel

Ausschließlich die Parent-Entität hat die Eigentumsrechte an der Child-Entität in einer Parent-Child-Beziehung. Wird die Relation gelöscht, so wird der Child-Eintrag automatisch mit gelöscht.

### Primärschlüssel propagieren

Wird der Primärschlüssel in einer Parent-Child-Beziehung propagiert, so wird die Primärschlüssel-Generierung für den Child-Eintrag ausgesetzt und der Primärschlüssel des Parents übernommen (wird z.B. häufig in History-Tabellen verwendet).

### Löschregel

Mögliche Werte:

- Nullify - Setzt die Child-Fremdschlüssel auf NULL im Falle, das der Parent-Eintrag gelöscht wird.
- Cascade - Löscht alle Child-Einträge mit dem Parent-Eintrag zusammen.
- Deny - Ein Parent-Eintrag kann nicht gelöscht werden, solange noch Child-Einträge über diese Relation gebunden sind.
- Do nothing - Parent-Einträge werden ohne Aktion auf der Child-Seite gelöscht. Dies kann im Zweifelsfall die Daten-Integrietät verletzen und sollte mit Vorsicht verwendet werden.

## Join Typ

Mögliche Werte:

- Inner Join – Quell-Datensätze werden nur für Relationen ausgegeben, deren Ziel-Datensatz existiert.
- Full Outer Join – Alle Quell-Datensätze werden angezeigt
- Left Outer Join – Quell-Datensätze ohne Ziel-Datensatz werden vor dem Löschen bewahrt
- Right Outer Join – Ziel-Datensätze ohne Quell-Datensatz werden vor dem Löschen bewahrt

## Quell Attribut

Das Attribut, von dem die Relation ausgeht

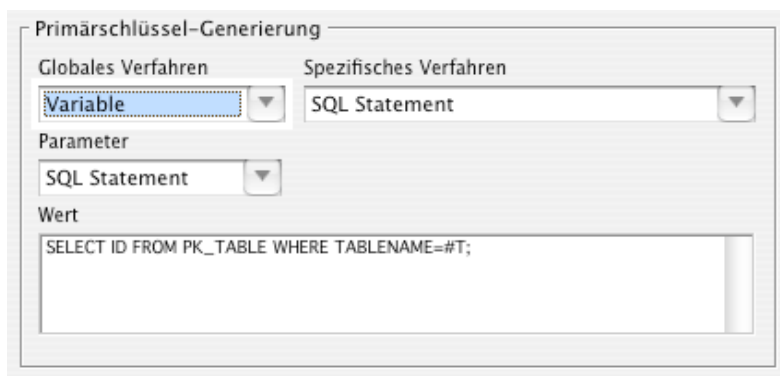
## Ziel Entität/Attribut

Entität und Attribut, auf die die Relation zeigt

## Batch Fehler Anzahl

Anzahl der gleichzeitig übermittelten Einzelfehlermeldungen im Falle eines Scheiterns des Auflösens einer ToMany-Relation. Das Nachladen von Folgefehlermeldungen erzeugt Last auf der Leitung zum Adapter, die somit minimiert werden kann. Die Auswertung der gesammelten Fehlermeldungen ist damit in die Business-Logik verschoben.

## Primärschlüssel-Generierung



### Globales Verfahren:

Einstellen des Verfahrens zur Primärschlüssel-Generierung: Auswahl des globalen Verfahrens (gilt Model-weit). Mögliche Einstellungen sind:

- Default – Es wird das Verfahren des JDBC-Adapters verwendet
- SQL Statement – Ein in „Wert“ zum Parameter „SQL Statement“ eingetragenes SQL-Statement dient als Primärschlüssel
- JavaClass – Es existiert eine Java-Klasse, die die Primärschlüssel-Generierung übernimmt
- Oracle Sequence – Für eine Oracle-Datenbank lassen sich Oracle Sequences zur Generierung einstellen
- Variable – Es ist global kein Verfahren zur Generierung hinterlegt, es greifen die spezifischen Verfahren.

### Spezifisches Verfahren:

Globales Verfahren ist variabel -> auf allen Ebenen lassen sich jetzt spezifische Generierungsverfahren einstellen:

Bei der Auswertung wird auf Attribut-Ebene angefangen nach dem Spezifischen Verfahren zu suchen, ist dies nicht gesetzt wird in der dazugehörigen Entität geschaut und dann im Model. Die Parameter zu dem Verfahren werden analog aber unabhängig vom Verfahren geprüft. Ist ein Parameter-Wert nicht gesetzt, so wird der Wert der nächsthöheren Hierarchiestufe angenommen. Findet sich abschließend kein Wert so greift das Standard-Verhalten des Adapters auch wenn spezifische Methoden eingetragen sind.

### Parameter

Mögliche Parameter sind :„SQL Statement“, „Java Class“ und für Oracle Datenbanken noch „Oracle Sequence“. Diese Parameter sind fest eingestellt für die dazugehörigen globalen Verfahren, sollte hier „Variable“ eingestellt sein so lassen sich Werte für alle Parameter setzen.

### Wert:

Hier werden die den Parametern zugeordneten Werte hinterlegt.

### Parameter SQL Statement:

Ein SQL-Statement, welches die Generierung des Primärschlüssels übernimmt. Es gelten folgende Platzhalter für das Statement: #T für die Tabelle, #C für die Tabellenspalte

**Parameter JavaClass:**

Eine Java-Klasse welche die Generierung des Primärschlüssels übernimmt. In dieser Klasse sind Schlüsselmethoden vorzuhalten, die vom System angesprungen werden. Eine Beschreibung des API's folgt

**Parameter Oracle Sequence:**

Ein Sequence-Name, welcher für die Generierung des Primärschlüssels zuständig ist. Es gelten folgende Platzhalter für den Sequence-Namen: : #T für die Tabelle, #C für die Tabellenspalte.